Lecture Notes in Statistical and Mathematical Methods for Artificial Intelligence 91255

Elena Loli Piccolomini
elena.loli@unibo.it

Academic Year 2020/2021

Last updated: September 2020

Thanks to: Giorgio Renzi

Lorenzo Mario Amorosa
Lucia La Forgia
Giacomo Pinardi

for the help provided

Contents

1 Linear systems

1.1 Introduction to linear systems

A linear system can be written as

$$\mathrm{A}\mathbf{x} = \mathbf{b}$$

where A is a matrix of size $m \times n$ (we suppose $m \geq n$) and $\mathbf{x}$ is a coulumn vector of length n and $\mathbf{b}$ is a column vectors of length m. $\mathbf{x}$ represents the unknown solution while $\mathbf{b}$ is a given vector. This form can be expanded as

$$\begin{gathered} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m \end{gathered}$$

We are interested in:

- Existence and uniqueness of a solution
- Numerical methods to find the solution
- Conditioning of the problem

We separately consider the case of square linear systems ($m = n$) and leasst squares problem ($m > n$).

1.2 Square linear systems

Proposition 1.1. The solution of a linear system

$$\mathrm{A}\mathbf{x} = \mathbf{b}$$

with A of dimension $n \times n$, $\mathbf{x}$ and $\mathbf{b}$ of dimension n exists and is unique iff one of the following conditions holds:

- A is non-singular
- rank(A) $= n$
- The system $\mathrm{A}\mathbf{x} = \mathbf{0}$ admits only the solution $\mathbf{x} = \mathbf{0}$

The solution can be algebraically computed in the following way

$$\mathrm{A}\mathbf{x} = \mathbf{b} \implies \mathrm{A}^{-1}\mathrm{A}\mathbf{x} = \mathrm{A}^{-1}\mathbf{b} \implies \mathbf{x} = \mathrm{A}^{-1}\mathbf{b}$$

The problem with this kind of method is that computing the inverse of A can be expensive for large values of n.

Another method to compute the solution of a system is *Cramer's rule*:

$$x_i = \frac{\det A_i}{\det A},\ i = 1, \ldots, n$$

where A_i is obtained from A by replacing the i-th column by $\mathbf{b}$

As for the previous one, the computation of determinants can be quite expensive for large values of n.

We will consider two different approaches to find the solution of a linear system:

- *direct methods*. They yield the solution in a finite number of steps; they are more precise but more expensive in terms of computational cost
- *iterative methods*. They require (in principle) an infinite number of steps; they are less precise and less expensive

It is very important to perform also an error analysis on the obtained results. It generally includes two parts:

- *Rounding or* arithmetic errors. They depend on the algorithm steps. An algorithm producing an arithmetic error limited by a constant is called a **stable algorithm**.
- *Inherent errors*. They are due to the errors in the data representation and the DO NOT depend on the algorithm. For this reason, if the inherent error is large,we speak about **ill-posed problem**.

1.3 Direct methods

In the case of direct methods we factorize the matrix A as in section **??**. Let A = LU , then the solution of the linear system $\mathrm{A}\mathbf{x} = \mathbf{b}$ can be computed by solving two triangular systems

$$\mathrm{L}\mathbf{y} = \mathbf{b} \text{ (forward substitutions algorithm)}$$
$$\mathrm{U}\mathbf{x} = \mathbf{y} \text{ (backward substitutions algorithm)}$$

Forward substitution.

Esempio 2.1 *Sostituzione in avanti.*

$$\begin{pmatrix} \ell_{1,1} & 0 & 0 \\ \ell_{2,1} & \ell_{2,2} & 0 \\ \ell_{3,1} & \ell_{3,2} & \ell_{3,3} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

$$\begin{array}{ll} y_1 = b_1/\ell_{1,1} & 1 \textit{ operazione} \\ \downarrow & \\ y_2 = (b_2 - \ell_{2,1} * y_1)/\ell_{2,2} & 3 \textit{ operazioni} \\ \downarrow & \\ y_3 = (b_3 - \ell_{3,1} * y_1 - \ell_{3,2} * y_2)/\ell_{3,3} & 5 \textit{ operazioni} \end{array}$$

Backward substitution.

$$\begin{pmatrix} u_{1,1} & u_{1,2} & u_{1,3} \\ 0 & u_{2,2} & u_{2,3} \\ 0 & 0 & u_{3,3} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

$$\begin{array}{ll} x_3 = y_3/u_{3,3} & 1 \textit{ operazione} \\ \downarrow & \\ x_2 = (y_2 - u_{2,3} * x_3)/u_{2,2} & 3 \textit{ operazioni} \\ \downarrow & \\ x_1 = (y_1 - u_{1,2} * x_2 - u_{1,3} * x_3)/u_{1,1} & 5 \textit{ operazioni} \end{array}$$

In the case of pivoting algorithm, we have that PA = LU. Then the solution of the linear system PA$\mathbf{x}$ = P$\mathbf{b}$ can be computed by solving two triangular systems

$$\begin{aligned} \mathrm{L}\mathbf{y} &= \mathrm{P}\mathbf{b} \text{ (forward substitutions algorithm)} \\ \mathrm{U}\mathbf{x} &= \mathbf{y} \text{ (backward substitutions algorithm)} \end{aligned}$$

1.4 Iterative methods

Iterative methods are numerical methods that approximate a solution using a procedure involving several (or an infinite amount of) steps.

The basic idea of iterative methods is to construct a sequence of vectors $\mathbf{x}_k$ enjoying the property of *convergence*

$$\mathbf{x}^* = \lim_{k \to \text{inf}} \mathbf{x}_k$$

where $\mathbf{x}^*$ is the exact solution and the starting guess $\mathbf{x}_0$ is given.

In general the sequence $\mathbf{x}_k$ is obtained as $\mathbf{x}_k = g(\mathbf{x}_{k-1}),\ k = 1, \dots$ where g is a particular function or set of operations acting on $\mathbf{x}_{k-1}$. Different classes of iterative methods are defined for different expressions of g. In general, an iterative method converges for matrices with fixed properties (such as on the shape, on the eigenvalues, ...).

Iterative methods are sensitive to both algorithmic and truncation errors.

Examples of iterative methods (the most common):

- *stationary iterative methods*, they take the form

  $$\mathbf{x}_{k+1} = \mathrm{B}\mathbf{x}_k + \mathbf{f}$$

  where B is called *iteration matrix* and $\mathbf{f}$ is a vector obtained from $\mathbf{b}$
- *gradient-like methods*, they take the form

  $$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$$

  where $\alpha_k \in \mathbb{R}$, $\mathbf{p}_k$ is a vector called *direction*.

  If A is symmetric and positive definite and the vectors $\mathbf{p_k}$ have the *conjugacy* property, i.e:

  $$\mathbf{p}_i^T \mathrm{A}\mathbf{p}_k = 0 \quad if\ i \neq k$$

  then the method is called *Conjugate Gradients*.

All these methods require one matrix-vector multiplication per iteration. Hence, if A is a *full* matrix the computational complexity is $O(n^2)$ per iteration.

Stopping criteria for iterative methods for linear systems are:

- defined the residual $\mathbf{r}_k = \mathbf{b} - \mathrm{A}\mathbf{x}_k$ at iteration k, stop the algorithm when
  - $\|\mathbf{r}_k\| \leq \epsilon$ (absolute criterion)
  - $\dfrac{\|\mathbf{r}_k\|}{\|\mathbf{b}\|} \leq \epsilon$ (relative criterion)
- $\|\mathbf{x}_{k+1} - \mathbf{x}_k\| \leq \tau$ (absolute criterion)

Sparse matrices A matrix A is called *sparse* if it has a very small percentage (i.e. $0.05 - 0.06\%$) of non-null elements. In this case, the matrix is not fully stored, but only the non-null elements and their indices are stored. The matrix-vector multiplication requires k multiplications, where k is the number of non-null elements.

1.5 Inherent errors in linear systems

WE remind that inherent errors are due to the errors in the data representation. They **do not** depend on the algorithm used for computing the result. Really, the analysis of inherent errors is performed by supposing to use exact arithmetic to compute the result.

Consider now a linear system $\mathrm{A}\mathbf{x} = \mathbf{b}$. What happens to the solution $\mathbf{x}$ when A and/or $\mathbf{b}$ slightly change (e.g. due to machine precision)?

Suppose that A doesn't change while $\mathbf{b}$ changes in $\mathbf{b}+\Delta\mathbf{b}$. The linear system becomes

$$\mathrm{A}(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{b} + \Delta\mathbf{b},\ \Delta\mathbf{x} \text{ inherent error}$$

We want to compare $\left|\left|\dfrac{\Delta\mathbf{x}}{\mathbf{x}}\right|\right|$ and $\left|\left|\dfrac{\Delta\mathbf{b}}{\mathbf{b}}\right|\right|$ to see how much the solution has changed with respect to $\mathbf{b}$. Let's subtract $\mathrm{A}\mathbf{x} = \mathbf{b}$ to $\mathrm{A}(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{b} + \Delta\mathbf{b}$

$$\mathrm{A}\Delta\mathbf{x} = \Delta\mathbf{b} \implies \Delta\mathbf{x} = \mathrm{A}^{-1}\Delta\mathbf{b}$$

Then we have

$$||\Delta\mathbf{x}|| = ||\mathrm{A}^{-1}\Delta\mathbf{b}|| \leq ||\mathrm{A}^{-1}||\ ||\Delta\mathbf{b}||$$

and from the linear system equation

$$||\mathrm{A}\mathbf{x}|| = ||\mathbf{b}|| \implies ||\mathbf{b}|| = ||\mathrm{A}\mathbf{x}|| \leq ||\mathrm{A}||\ ||\mathbf{x}|| \implies ||\mathbf{x}|| \geq \left|\left|\frac{\mathbf{b}}{\mathrm{A}}\right|\right|$$

Thus we have

$$\left|\left|\frac{\Delta\mathbf{x}}{\mathbf{x}}\right|\right| \leq ||\mathrm{A}^{-1}|| \cdot ||\mathrm{A}|| \cdot \frac{||\Delta\mathbf{b}||}{||\mathbf{b}||} = K(\mathrm{A})\frac{||\Delta\mathbf{b}||}{||\mathbf{b}||}$$

Definition 1.1. The *condition number* of a matrix $\mathrm{A} \in \mathbb{C}^{n\times n}$ is defined as

$$K(\mathrm{A}) = ||\mathrm{A}||\ ||\mathrm{A}^{-1}||$$

where $||\cdot||$ is a matrix norm. In general $K(\mathrm{A})$ depends on the choice of the norm, indicated by a subscript.

Notice that $K(\mathrm{A}) \geq 1$ since

$$1 = ||\mathrm{A}\mathrm{A}^{-1}|| \leq ||\mathrm{A}||\ ||\mathrm{A}^{-1}|| = K(\mathrm{A})$$

The condition number measures how sensitive a function is to changes/errors in the input and how much the output changes as a result of the operations performed. A *well-conditioned* system is a system where $K(\mathrm{A})$ is small, while an *ill-conditioned* system is a system where $K(\mathrm{A})$ is large.

If $K(\mathrm{A})$ is very large, the matrix A is near (with respect to a norm) a singular matrix and its columns are quasi-linearly dependent. Regularization techniques can be used in order to reduce $K(\mathrm{A})$.

Example 1.1. The condition number for p-norm with $p = 2$ is

$$\begin{aligned} ||\mathrm{A}||_2 &= \sqrt{\rho(\mathrm{A}^T\mathrm{A})} = \sigma_{max} \\ ||\mathrm{A}^{-1}||_2 &= \frac{1}{\sqrt{\rho(\mathrm{A}^T\mathrm{A})}} = \frac{1}{\sigma_{min}} \\ K_2(\mathrm{A}) &= ||\mathrm{A}||\ ||\mathrm{A}^{-1}|| = \frac{\sigma_{max}}{\sigma_{min}} \end{aligned}$$

where λ_{max} and λ_{min} are the maximum and minimum singular values of A

1.6 Linear least squares

Consider an *overdetermined system*

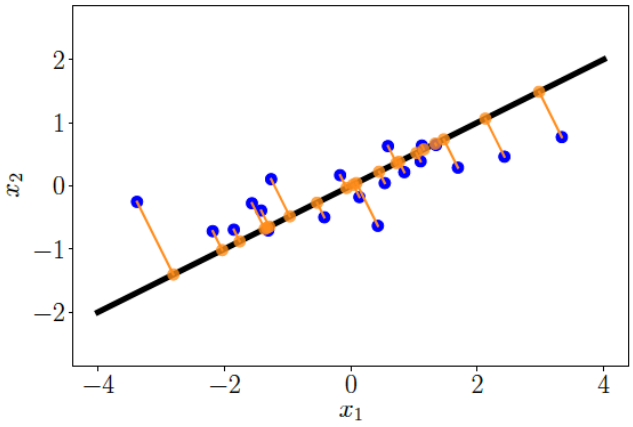
$$\mathrm{A}\mathbf{x} = \mathbf{b}$$

where A has dimension $m \times n$ with $m > n$, $\mathbf{x}$ of dimension n and $\mathbf{b}$ of dimension m. Such a system usually has no solution, meaning that $\mathbf{b}$ does not lie on the subspace spanned by the columns of A (denoted by Col(A) from now on). Since no exact solution exists for this problem, we would like to compute the best approximate solution. We need some tools.

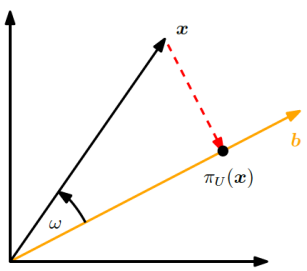
1.6.1 Orthogonality and projections

Projections are an important class of linear transformations. In machine learning one often deals with high-dimensional data which is hard to visualize

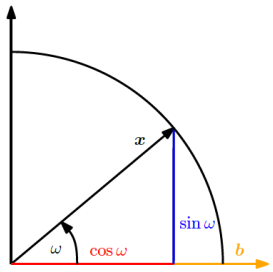
and analyze. Oftentimes, only a few dimensions contain useful information, meaning that other dimensions are not essential to understanding the data. When applying data compression techniques we want to minimize the loss of information by finding the most informative dimensions of the data first.



Orthogonal projection onto a subspace of dimension 1.



(a) Projection of $x \in \mathbb{R}^2$ onto a subspace U with basis vector b.



(b) Projection of a two-dimensional vector x with $\|x\| = 1$ onto a one-dimensional subspace spanned by b.

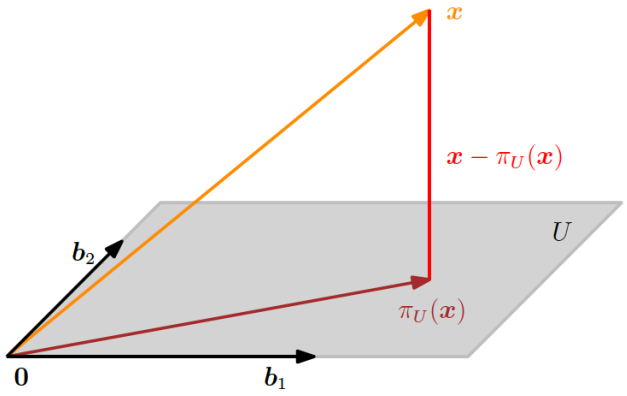
- The projection $z = \Pi_U(\mathbf{x})$ is the closest point to $\mathbf{x}$ of the subspace U generated by $\mathbf{b}$ by considering as the measure of the distance the 2-norm ($\|\cdot\|_2$). The vector $\mathbf{z} - \mathbf{x}$ (the red dashed one in the figure) is orthogonal to the subspace U (and to the vector $\mathbf{b}$ generating U). Hence, the scalar product: $< \mathbf{z} - \mathbf{x}, \mathbf{b} >= 0$
- The vector $\mathbf{z}$ is a vector of the subspace generated by $\mathbf{b}$, hence $\mathbf{z} = \alpha\mathbf{b}$, $\alpha \in \mathbf{R}$.

We can find α in the following way

$$\langle \mathbf{x} - \mathbf{z}, \mathbf{b} \rangle = 0 \implies \langle \mathbf{z} - \alpha \mathbf{x}, \mathbf{b} \rangle = 0 \implies \langle \mathbf{x}, \mathbf{b} \rangle - \alpha \langle \mathbf{b}, \mathbf{b} \rangle = 0$$

$$\alpha = \frac{\langle \mathbf{x}, \mathbf{b} \rangle}{\langle \mathbf{b}, \mathbf{b} \rangle} = \frac{\mathbf{x}^T \mathbf{b}}{||\mathbf{b}||^2}$$

Example when projecting from $\mathbf{R}^3$ to $\mathbf{R}^2$.



We now generalize to sub-spaces of dimension possibly greater than one.

Theorem 1.1. *(Best approximation theorem) Let W a subspace with dimension p of $\mathbb{R}^n$, $\mathbf{y} \in \mathbb{R}^n$ and $\hat{\mathbf{y}} = \text{proj}_W(\mathbf{y})$. Then $||\mathbf{y} - \hat{\mathbf{y}}|| < ||\mathbf{y} - \mathbf{v}||$, $\forall \mathbf{v} \in W$, $\mathbf{v} \neq \hat{\mathbf{y}}$.*

This means that $\hat{\mathbf{y}}$ is the best approximation of $\mathbf{y}$ in W.

Example 3.10 MML book.

The idea is to use projections to find the vector in Col(A) (i.e. the subspace generated by set of all the columns of A) which is closest to $\mathbf{b}$. As already seen in the previous chapter, this vector is indeed the orthogonal projection of $\mathbf{b}$ onto the subspace Col(A).

1.6.2 The linear least squares problem

When A has size $m \times n$ with $m > n$, it is not possible to find a solution of the linear system $A\mathbf{x} = \mathbf{b}$. The orthogonal projections allow to find a vector $\mathbf{x}$ that is *near* the system solution (in the sense of the 2-norm).

The problems is then formulated as to compute $\hat{\mathbf{x}}$ so that:

$$\hat{\mathbf{x}} = argmin_x \|\mathrm{A}\mathbf{x} - \mathbf{b}\|_2^2 \tag{1}$$

For what concerns existence and uniqueness of the solution of the linear least squares problem, two different cases are possible:

- $\mathrm{rank(A)} = n$, meaning that every column of A is linearly independent. A unique solution exists $\forall \mathbf{b} \in \mathbb{R}^m$. Let $\mathbf{x}^*$ be the approximate solution to the linear system, we have

  $$\mathrm{A}\mathbf{x}^* = \mathrm{proj}_{\mathrm{Col(A)}}(\mathbf{b})$$

  This means that

  $$\mathbf{b} - \mathrm{A}\mathbf{x}^* \in [\mathrm{Col(A)}]^{\perp} \implies \mathrm{A}^T(\mathbf{b} - \mathrm{A}\mathbf{x}^*) = 0$$

  Thus, we obtain the *normal equations*

  $$\mathrm{A}^T\mathrm{A}\mathbf{x}^* = \mathrm{A}^T\mathbf{b}$$

  which is a linear system in n equations and n unknowns. $\mathrm{A}^T\mathrm{A}$ is called *Gramian matrix* of A and is non-singular, positive definite and symmetric. The approximate solution can then be obtained

  $$\mathbf{x}^* = (\mathrm{A^T A})^{-1}\mathrm{A}^T\mathbf{b}$$

- $\mathrm{rank(A)} = k$, $k < n$, meaning that there is an infinite number of solutions (∞^{n-k}). We can find the least squares solution to the problem by using the pseudoinverse of A obtained via SVD

  $$\mathrm{A}\mathbf{x} = \mathbf{b} \implies \mathrm{A}^+\mathrm{A}\mathbf{x}^* = \mathrm{A}^+\mathbf{b} \implies \mathbf{x}^* = \mathrm{A}^+\mathbf{b} \implies \mathbf{x}^* = \sum_{i=1}^{k} \frac{\mathbf{u_i}^T\mathbf{b}}{\sigma_i}\mathbf{v}_i$$

It is possible to modify the linear least squares problem (1) by introducing a weight matrix with the aim to give different weights to the components of $\mathbf{x}$.

$$\hat{\mathbf{x}} = argmin_x \|\mathbf{W}\mathring{A}\mathbf{x} - \mathbf{b}\|_2^2 \tag{2}$$

where $\mathbf{W}$ is an invertible matrix. In this case the *weighted normal equations* are:

$$(\mathrm{WA})^T\mathrm{WA}\mathbf{x} = (\mathrm{WA})^T\mathbf{b}$$